Accessibility Pi/O

Carlos Armendariz, Ji Chang, and Jorge Tamayo
Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—Our goal is to create an inexpensive, open-source keyboard and mouse set accessible to those with cerebral palsy. Most dedicated accessibility I/O tech currently on the market requires the user to assemble it themselves, and prebuilt one-handed keyboards are much more expensive than standard keyboards.**

***Index Terms*—Accessibility, Cerebral Palsy, Keyboard, Mouse**

1 INTRODUCTION

Cerebral palsy (CP) is an umbrella term for a group of diseases that impair a person's motor functions. Spastic hemiplegia is one such form of CP, in which one side of the body is impacted much more severely than the other. Since modern computer use assumes the user has full mobility of both hands, those who can only reliably use one side of the body are forced to either use tech not designed for them, assemble their own, or purchase expensive specialized keyboards.

Our goal is to help those with Cerebral Palsy by designing inexpensive I/O hardware that is operable with one arm and one leg. Current solutions are either more expensive than standard I/O or require additional hardware to achieve full use, and many current accessibility input devices for computers are not designed to allow simultaneous mouse and keyboard use. We aim to create a design that will not be as broadly applicable to all motor disabilities, but will be a practical tool for those with spastic hemiplegia specifically. This would consist of a one-handed keyboard and a mouse operable by foot, both of which connect to a programmed Raspberry Pi that translates the inputs for a host computer.

To keep the design accessible to everyone, the project will be open source and use inexpensive, ubiquitous components. This allows people to easily and cheaply reproduce our results, or improve upon them if need be.

2 USE-CASE REQUIREMENTS

The keyboard should allow a baseline level of proficiency in typing; it is only worth using if someone can type quickly on it. Our goal is for users to type about 30 WPM once acclimated.

Also, to be usable for most tasks, a certain number of characters must be supported. A total of 74 characters is the baseline we will aim to meet, as this will allow most tasks to be performed on a computer. This includes 52 upper/lowercase letters, 10 digits, space, enter, and punctuation ('.,";()[]?!). Also, to be usable, other supporting keys need to be present. The non-character keys we plan on adding are: spacebar, left/right mouse click, backspace, enter, shift, ctrl, alt, a toggle key for hold key presses (i.e. if you press and hold a key, it will only read one press), and a toggle key to type digits (a la Num Lock).

We also aim to make a comfortable design. The keyboard should be easy to use without much strain. The design itself also needs to be durable; if it breaks within a week of use, then it is not a very useful design. So for our purposes, the switches used need to function after a large number of presses, and the design itself should be able to withstand some abuse.

The keyboard should also be able to interface with a text to speech program in addition to standard text entry. This is because people with cerebral palsy can sometimes have issues with the fine motor control required for speech. Also, the keyboard and mouse should be able to be used simultaneously. This is a key requirement for anyone who wants to play games or use certain CAD tools.

The quantitative metrics that we aim to test are how well it reduces the rate of common errors while typing. The most common type of error found in people with disabilities (at a rate of about 10%) was a long key press error [2]. This occurs when the user holds a key for longer than intended after pressing it, resulting in extra characters being input. Long key press errors were also found to be a non-issue for people without disabilities. Other common errors in people with disabilities were additional key press errors, missing key errors, dropping errors, and bounce errors. We aim to create a design that implements hardware and software features that can practically reduce the rate of these errors to match the occurrence in people who have full motor functionality.

The latency on the keyboard and mouse are also important. Too much delay can render an input device unusable, so any solution that introduces excessive latency should be avoided.

Our final requirement is that the production cost for the final devices should be below $200. This is to make it more accessible to people with disabilities; most purpose-built devices are significantly more expensive than this, and the cheaper variants usually require additional hardware and assembly for full functionality.

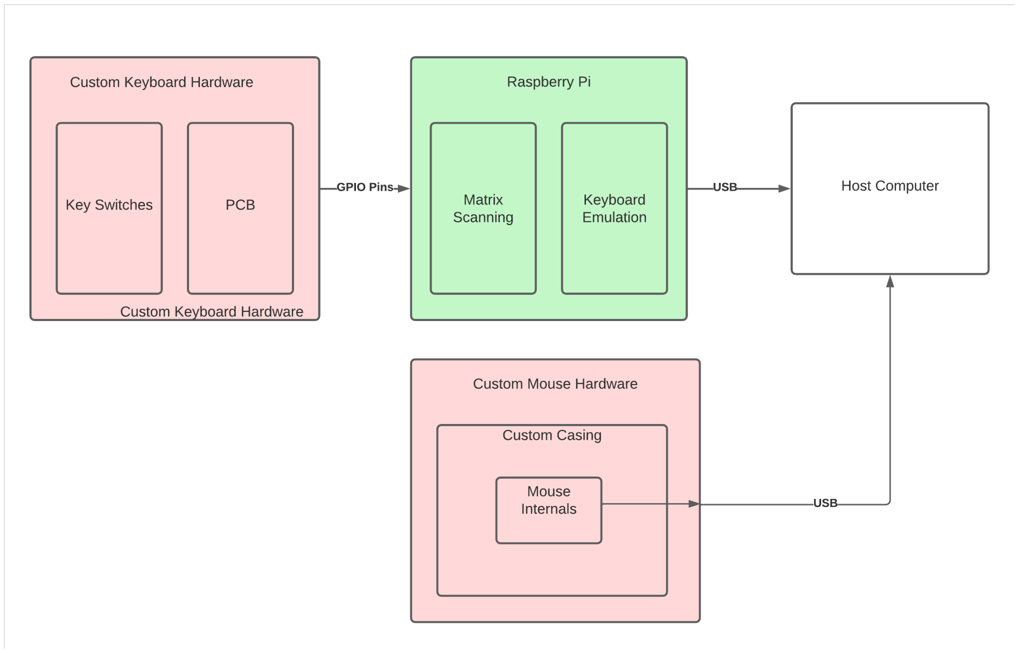

Figure 1: Our system block diagram.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

Our system works first by subdividing the tasks into the keyboard, mouse, and Raspberry Pi.

The GPIO pins on the Raspberry Pi receive the key-press signals from the keyboard. Since a large number of keys are required, we will employ a technique called matrix scanning, which is commonly used for other keyboards. Matrix scanning allows keys to be detected in a row and column format as if they were laid out in a grid. Usually, the physical layout is a grid, but this is not a requirement for matrix scanning. So, a small number of GPIO pins can be used to detect a large number of keys. For example, if eight pins were used, 16 keys could be distinguished from each other (four rows and four columns).

So Matrix Scanning allows the Raspberry Pi to determine when and which keys are pressed. It then sends the corresponding key over USB to the host computer.

Matrix scanning works by actively setting columns to be high or low, and reading the resultant rows. The Pi scans through the columns by setting every column except for one high, and rapidly switching which column is low. If a key is pressed, it closes a switch, which pulls the entire row low to match the column (the Pi cycles through the columns so rapidly that it can be assumed that the column is low at any given time). The mouse uses USB input as well.

Our keyboard also contains five keys that can change how the Pi reads the input: shift key, a num lock, ctrl, alt, and a hold press lock. The first two are meant to allow the same key to be mapped to multiple characters; shift turns lowercase into uppercase and num turns certain letters into numbers. Shift works like a phone keyboard, in that if pressed, it will only capitalize the first letter pressed. The num lock will work like a normal lock key, where pressing it on will turn all the letters into numbers until it is pressed off. Num lock will override shift. Alt and Ctrl are also held keys.

The last key, the hold press lock, is meant to help with the long key press error (when the user presses a key longer than they mean to, resulting in a longer string of letters). While the hold press key is off, pressing and holding a key will only count for one press, and while the hold press key is on, holding the key will be treated as multiple presses. We give the user the option of turning the hold press on and off because there are some cases where repeated presses are desirable, such as playing a video game. Like num lock, pressing the hold press lock once will turn it on until pressed again.

4 DESIGN REQUIREMENTS

For ergonomic purposes, the individual keys on the keyboard can be increased in size relative to that of a standard keyboard. Increasing the size of and distance between keys is a common fix for people with dexterity issues. However, the key size is limited by how fast we expect people to be able to type; if the keys are too big or are spaced too far apart, then typing speed can suffer. A size that appears to be a good balance is a minimum of 20mm on both sides of the key cap, approximately 30% larger than usual. We will also be using linear key switches because they are easy to make and comfortable to press.

Due to the limited number of GPIO pins on a Raspberry Pi, and the total number of characters that we want to support (74 at minimum), the keyboard circuit is a matrix circuit. This determines which key is pressed by actively scanning the columns and seeing which row “reacts”.

Diodes prevent unwanted current paths and ensure that when a key is pressed, the row voltage is equal to the diode’s forward voltage. Since the Raspberry Pi reads any voltage above 1.8V as a high digital signal, the forward voltage on each diode must be less than 1.8V. We need a diode for every key switch on the board.

Since the computer will see our device as a standard keyboard, the requirements for being able to interface with a text-to-speech program will come for free. Any keyboard is usable for these programs, so ours will be no different in this area. In order to reduce the rate of common errors while using the keyboard, a combination of hardware and software solutions will be employed. There will likely need to be software that recognizes when a key is being held unintentionally in order to avoid long key press errors. Also, a removable key guard will likely be required to prevent missed key errors.

To keep latency low, there are a limited number of actions we can perform in software between detecting a key press through matrix scanning and sending the key press to the computer. The exact limit here will require testing.

5 DESIGN TRADE STUDIES

5.1 Designing a Mouse Alternative

For our implementation of mouse functionality, we had many considerations to make. Most current solutions involve a mouse alternative such as a ball mouse or a roller mouse, or for the Microsoft adaptive controller, a large joystick to control mouse functionality. We went over many iterations of ideas but ultimately decided that these solutions were not in line with what we wanted to implement.

All of these solutions have the added downside that the control of a cursor would be very complicated and would require very difficult movements to use. We wanted someone to be able to preserve the same precision as a typical mouse. This compounded with our second criteria, which was that we wanted the mouse and the keyboard to be usable at the same time. It is a simple feature but one that is not typically seen.

We opted then to do a foot controlled mouse, which we felt was a good alternative since we maintain the finer controller of a typical mouse, without sacrificing the one hand use we planned to implement for the keyboard.

5.2 Designing a Keyboard for a One Handed User

The general layout of our design was going to be a one handed keyboard. We felt this was an easier and more practical decision than designing a keyboard for two handed use in which one hand’s mobility is limited.

Designing a one handed keyboard from scratch is likely impractical, as one handed layouts already exist and have likely not only been established for much longer, but also been more thoroughly tested than any completely new layout we could come up with. Considering this, we decided to base our custom layout on a pre-existing design, but modifying certain aspects to make it more useful for our particular user and use-case.

By designing a keyboard that is meant to be used with one hand, we also felt we create the potential for someone to gain good typing speed with the device, despite having SH.

5.3 A Keyboard for CP and SH

With the general design of the one handed keyboard already established, we needed to find a way to attend to the particular difficulties of someone with CP. We went through sources of people with CP to try and gauge what the typical difficulties with CP, and more specifically SH, were. Among these issues were keys being too small or being placed too closely together. While these might seem like relatively minor issues, for someone with SH these issues are very real and are the most common cause for accidental error input.

These are the particular considerations that were kept in mind and should be addressed when designing the ergonomics of our device. These are the primary inspirations for our choice to use 20mm key caps and including a key guard in our design. Our feedback from Professor Carrington also confirmed the importance of these elements in our design.

We also decided on linear key switches as they are the easiest kind of key switch to press down on. Other key switches include tactile and click switches. We opted to go with linear as opposed to these because linear key switches have smooth travel as they are pushed down, while the others have a kind of push-back part way through the motion.

5.4 Possible Further Considerations

One thing we did not consider when making our design, but may be useful when considering economics in the future, are the parts of the body which we are not designing for. Our feedback from Professor Carrington gave us much insight into the kinds of details we might have overlooked.

For one, although our design is intended to be used for the side of the body that is high functioning, that does not necessarily mitigate the problems present from the other side. The less functional side of the body is prone to spastic behavior which can affect the comfort and ease of use of the side of the body that is high functioning through indirect means. We can resolve this issue likely through having a more considerate picture of the kind of person we are trying to create this device for and stresses the importance of finding a person with CP to consult.

6 SYSTEM IMPLEMENTATION

6.1 Keyboard PCB Specification

In Figure 2, the matrix scanning circuit is shown. To determine which keys are pressed, an individual column is set to low by the Raspberry Pi. Then the value on each row is read. If the row has a value of 0 (less than 1.8V across the diode), then the key in the column that is set low and that row is pressed.

Matrix scanning works by actively setting columns to be high or low, and reading the resultant rows. The Pi scans through the columns by setting every column high except for one; the column set low is the one being scanned. The Pi scans through each column faster than people can detect, a mechanism typically used in keyboards.

If a key is pressed, it closes a switch, which pulls the entire row low to match the column (the Pi cycles through the columns so rapidly that it can be assumed that the column is low at any given time). This means that for n pins we get $(\frac{n}{2})^2$ possible inputs.

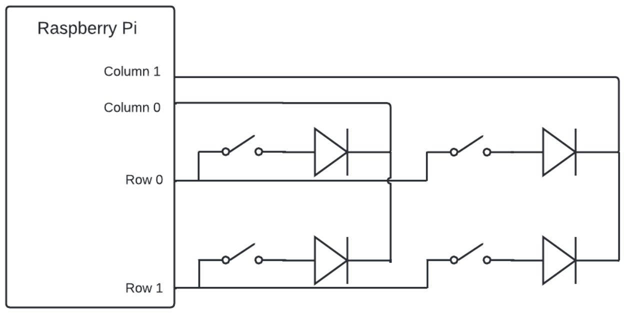

Figure 2: Matrix scanning circuit diagram[1].

The diodes on the keyboard help to stop current from flowing in a direction that isn't desirable. While technically they are optional, most if not all standard keyboards include them because it makes them less prone to error. Since diodes are imperfect and have forward voltages, pressing a key sets the row voltage equal to the diode's forward voltage. This is why diodes should have forward voltages less than 1.8V, the Pi's threshold between high and low voltage.

The PCB is a fairly standard matrix scanning circuit as mentioned prior, with spacing between the switches allowing for our non-standard 20mm key caps. The outputs of the keyboard are then fed to the Pi by way of male header pins. The Pi also includes male header pins and we plan to have a female-female cable connecting the Pi to the Keyboard PCB

6.2 Typing Behavior

The keyboard operates in a slightly non-standard way due to it being a one-handed keyboard, as well as having some particular features which makes typing for people with disabilities much easier.

The typical combination keys (num, shift, fn, ctrl, etc.) are modified by our software to behave in a way that is atypical for a keyboard.

As an example: when the shift key is pressed, it will toggle on, if pressed again, it will toggle off. If the shift key is toggled and a letter is pressed, it will be typed as a capital letter and then shift will be automatically detoggled.

This allows for more comfortable one handed use of a keyboard while also being a particular benefit to those with SH. The keyboard also includes keys to replace left right click, which is also typical for one-handed keyboards. This feature complements our mouse alternative which only performs the cursor tracking function of a mouse.

The implementation of the keyboard saw some changes throughout our design but we stuck to it fairly consistently. The main change was key layout, which saw some improvements made after referencing one-handed keyboard research; we used right-handed Dvorak as a reference. The overall button positions and keyboard shape stayed the same however. The keyboard also has a hold-toggle key as we intended to remove hold errors but the auto-detoggling keys feature was not implemented. The extra key we allocated to the special feature was used for hold toggle because it seemed more relevant to our target group. Whilst we could implement auto-detoggling, we did not see it as a high priority, The num-toggle also allows the use of the shift key without needing to hold it, effectively turning it into a caps lock.

6.3 Keyboard Ergonomics

The Ergonomics of our Keyboard is largely based on the typical complaints from people with SH. Although people with SH are capable of using both hands, the one that is affected is seldom used, and most are relegated to typing with one hand on a two handed keyboard.

We found that people with SH and CP tended to prefer keyboards with larger key caps because it was much easier for them to type. A common error was also the accidental pressing of multiple keys, which people with SH often tolerate but dislike. The feature of a key guard is not often seen in keyboards made for adults, and is something that people with CP often prefer. Our keyboard ended up having extra spacing between keys to add room for a key guard. This allows our target audience to use it more easily. The key guard is optional and easily removable in case the user would prefer to type faster.

6.4 Mouse Implementation

The mouse will operate as a typical USB mouse but with fewer features. Since left right click are relegated to the keyboard, the mouse only needs to move the cursor. Since the user should be able to use the mouse and keyboard simultaneously, the mouse will be controlled with one foot.

The implementation of the mouse is fairly simple. We will purchase a mouse and replace the housing with our own custom design, made to be strapped to the underside of a flat shoe. While implementing an entirely custom mouse PCB was an option, it seemed impractical, especially since our mouse does not have any unique features apart from its unique ergonomic design

All in all, the mouse implementation worked surprisingly well. The mouse strapped to one foot provides some finer control to that of other possible solutions while also being fairly simple and consistent. The primary drawback was are inability to tweak our design when taking spastic behavior into consideration. It is possible that for a mouse strapped to the high functioning side of the body, that use is impeded by spastic behavior of the opposite leg. Overall, we were pleased with the solution for its level of practicality.

7 TEST & VALIDATION

When it comes to testing an input device, there are some fairly common problems that arise. We hope to be attentive to these considerations and build a thorough testing plan for our device. We are limited by the availability of people with CP to test the device with, which could be a potential shortcoming in the future.

Regardless, we have tried to implement a survey based testing plan to try and gauge the quality of our device from comfort and ergonomics to ease of use and error prevention.

The verification testing in this context would be seeing if the keyboard and mouse combination is functional, and validation testing is checking if people with CP can use our keyboard.

7.1 Tests for Determining Error Reduction

Since we could not find any test volunteers within our demographic, we could not check if our design reduced errors or not. Also, since our design is unfamiliar, any error improvements may not be because of our design; rather, since testers are not touch-typing and instead look at the keyboard, they type much more carefully.

We found a common error where pressing the spacebar sometimes produced a double space as it did not lay on the lower portion of the keyboard like a typical spacebar. Also, there was a similar error with ! and . In that case, pressing ! sometimes added a . by accident (like !.).

7.2 Survey Goals

To test our design, we created a survey and found volunteers to test out our keyboard by typing both complex and simple sentences. The purpose was to test the layout and determine what level of confidence people had with our keyboard. We did not expect people to be great at using it to begin with, as it would be too unfamiliar, but we asked them to rate their confidence about whether they could get used to using this keyboard given time and practice. We also asked them if they had any issues with particular keys or improvements they feel they would want.

In general, results were positive with people feeling confident they could learn our layout with some time and practice. Some layout changes could be made to improve comfort such as moving the auxiliary keys more towards the left side or shifting the space bar to be lower, like a typical keyboard.

7.3 Survey: Can it be learned?

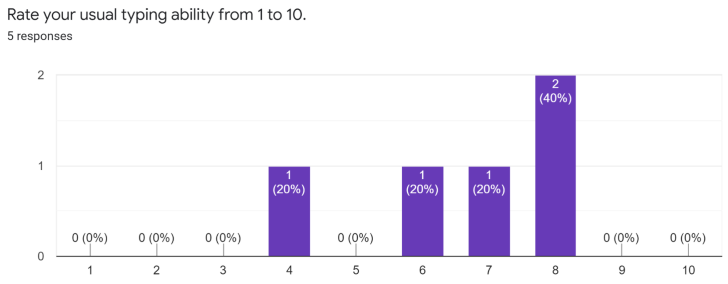

Figure 3: Testers' rating of their typing ability on a standard keyboard

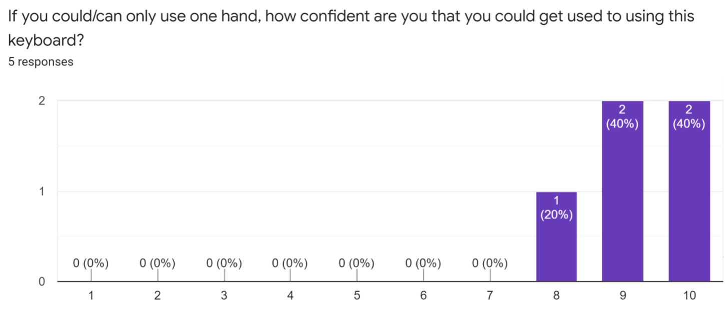

Figure 4: Testers' Confidence of their ability to use our keyboard with one hand.

People seemed to generally have an interesting confidence in the new layout. Our layout is based on right-handed dvorak, an alternative to qwerty that allows for faster typing with one hand. Our testers took interest in the layout and felt that if they had gotten used to it they could potentially type very fast. Words like "the" were particularly easy to type on this keyboard, and the potential of the layout was often cited as a primary source of confidence for the design.

7.4 Survey: Potential improvements

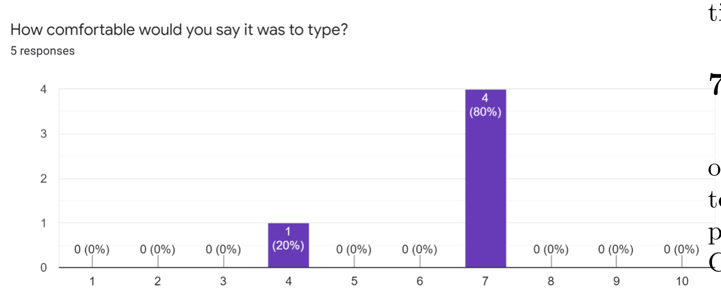

Figure 5: Testers' rating of their comfort using our keyboard.

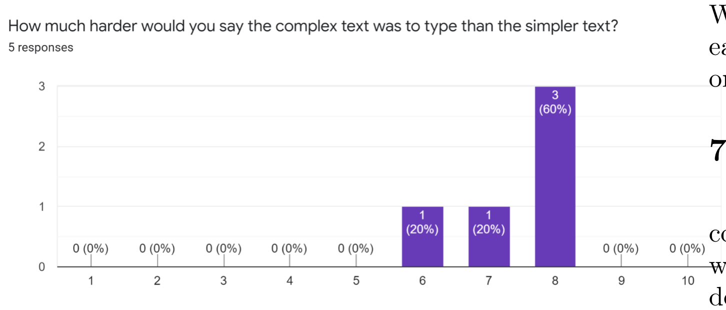

Figure 6: Testers' rating of the difficulty between complex and simple texts.

Some keys were particularly awkward to use, so we would change them if given the chance to redesign. The relative positions of letters and the general layout, as stated previously, was received favorably, but special keys at the edges of the keyboard somewhat less so. The space bar could have been less awkward to use if it had not rested above two keys, and some of the keys on the right side (such as alt and ctrl) were harder to reach. These conclusions were based on the qualitative questions we had at the end of our survey:

1. Are there any mistakes you encountered while typing? (Example: Did you press a key you were not intending to?)

2. Were there any inputs in particular that you felt were too difficult to perform? For example, was it more difficult than necessary to press the comma key?

3. Do you have any suggestions for improving the layout?

The special hold toggle key we implemented also did not have to be situated as close as it was, as it was not a key that would need to be pressed often.

Other potential redesign features would address user feedback, including LEDs indicating on/off for toggle keys and software features for error reduction.

The layout had also changed from our preliminary design. This was, given the responses, a significant improvement. The older layout (which was not based on research) had its limitations. The new one-handed dvorak that was designed has much potential and given thorough testing could prove to be a very efficient design for typing at relatively high words per minute.

7.5 Test Limitations

The limitations of our tests come largely from the lack of success in finding people from our target demographic to test with. In the final weeks it became difficult to find people willing to test our design and were limited to a few Carnegie Mellon students.

The tests allowed us to verify at the very least the potential of our design and we were able to verify through personal use as well as through our testers that it was not only possible to operate a computer with one side of the body but that it was actually comfortable in many ways. Whilst it was certainly difficult for some, it was surprisingly easy for others and it is apparent that with some practice one could easily get used to the device.

7.6 Verification Validation

The foot mouse we designed allowed people to have fine control of the cursor and people were able to play Minecraft with relative ease despite only having just begun to use our device. This meets the requirements we set out to overcome as other mouse alternatives tended to lack the fine control that we could provide with a relatively simple solution.

The keyboard also meets many of our requirements. Whilst a few minor redesigns would be beneficial, we did manage to create a keyboard that could be used with one hand and which had the potential for some very fast typing and which had effectively no more latency than a typical keyboard. The keyboard also functioned as a mouse as that is where the left and right click inputs were. The main aspect of the keyboard we were unable to verify was the error reduction, because we could not find testers within our target demographic.

We saw no glaring issues during testing but it would have been better, given more time, to have a group from our target demographic test it thoroughly and provide feedback for a redesign. This would bolster our ability to verify and validate our design. Otherwise, the features we implemented and the housing design matched what we found people with CP often found favorable. Without direct testing however, it is not known whether these features are as beneficial as they are intended to be.

8 PROJECT MANAGEMENT

8.1 Schedule

Our plan for implementing our accessibility based input device begins mainly with the keyboard itself. In general, whatever hardware we make has a complementary piece of software which can be worked on in parallel. So the first step of implementing the software and hardware of the keyboard is the first step. What follows is a design phase, in which the practical aspects of our design have already been thought through and we need only to deal with the physical ergonomics of what we make.

At this stage we begin work on the mouse input device, which is less complex and requires only the design of a new housing for mouse internals. This is why it is part of the design phase and not the initial hardware and software implementation phase.

Our schedule did change from our original. The software has become somewhat more complicated than expected, primarily due to the non-standard behavior we were trying to implement, along with dual functionality of mouse and keyboard. The software was essentially being edited from the start of the intended beginning stage, all the way to the integration step at the end of the semester. Not only that, but delays in having our board shipped and a week when we were all sick pushed back a lot of the work we were intending to do. We also had a section scheduled for testing which was greatly reduced as we were unable to find people to test our device. A lot of work on the housing got done far closer to the end than we had intended, but all major features were luckily implemented in time.

8.2 Team Member Responsibilities

There are mainly four different aspects to our division of labor: hardware, software, design work, and design implementation. For Hardware, Jorge and Ji are primarily responsible. The software has typically been Carlos' responsibility, but we are all capable of working on it. The design work is a shared collaboration between everyone and the actual implementation (3D printing, and construction) will also be a shared responsibility.

Whilst our responsibilities did not change drastically, since the software had lasted much longer than expected, Carlos had to work on that for a longer period. Jorge had the circuit and PCB design as his focus and Ji did research on optimal keyboard layouts and housing. In the end we all worked together to actually build the housing which we had designed.

8.3 Bill of Materials and Budget

Our budget of six hundred dollars is more than enough to implement what we are planning to make. The highest cost item will likely be the custom PCB, which we plan to order from Oshpark. This board will likely cost a significant amount as it is quite large and there will likely be a minimum order number of at least three boards.

Our bill of materials changed a lot since our design review. We managed surprisingly to find a relatively cheap option for getting the boards built despite them being quite large with a total cost of \$15 for 5 whole boards. We did spend more money on miscellaneous materials though for constructing the housing for our mouse and keyboard, such as rubber tape to prevent the board from moving while typing. We also needed to purchase a large mouse pad so that our foot controlled mouse would be usable.

8.4 Risk Management

Our primary risk is implementing an input device that is unhelpful. We have mitigated this already by meeting with a specialist in accessibility devices and are making active plans to seek out and consult someone with the relevant target disability. The more logistical risks of a custom PCB not working, can be mostly mitigated by prototyping our keyboard circuit before committing to ordering the board. Other parts such as the linear key switches can have delays, but that risk does not seem very likely. All our components, even the Diodes are available through many sites, even amazon.

Our PCB design worked quite well and we did not even need to revise it, so that saved us some time as we did not need to revise the PCB layout.

One potential issue came in the risk of time management as unforeseen issues like shipping delays and our team being sick pushed back much of our work. Ultimately we managed to work the extra hours to catch back up as much as possible but it did have drawbacks. Part way through the semester we considered a mouse redesign but we ended up needing to fall back on our first design due to a relatively simple implementation plan which we could get done on short notice. The mouse design was luckily very practical and comfortable to use.

The key caps we ordered also lacked mounting prongs and were not ideal for using on our keyboard but it was too late into the semester and we needed to do the soldering. This resulted in keys that were positioned somewhat unsteadily, which we would improve given more time.

9 ETHICAL ISSUES

Primary ethical issues involve a lack of consideration for nuanced issues. When we began, we did not consider how the spastic behavior of one side of the body could affect the high functioning while the keyboard is in use. This means that our device could have some design problems we were unable to consider and as such could present ethical issues through a misrepresentation of the problem. This means that any market solution runs the risk of creating a platform that people wont desire using but will also discourage others from attempting to create a better system. For example, currently in the Computer accessibility market the focus is on modularity which matches a general demographic but lacks the ability to focus on specific issues.

Social expectation then is a big factor. It would be a shame if our solution would be seen as sufficiently viable that the problem is no longer considered, especially when the solution is flawed. This becomes an even greater issue when we consider our unfortunate inability to find a proper demographic to test it with. This creates an inherent misrepresentation in our project as we tried to solve issues that we found were typical and frustrating for those with SP mainly through research with little contact with

Table 1: Bill of materials

Description	Quantity	Cost Per Item	Total
Custom PCB	5	$3	$15
Diode - 1N4148	100	$0.05	$5
Mouse Internals	2	$10	$20
3D Printing	n/a	n/a	$50
Misc. Materials	2	20	$40
Mouse Pad	1	15	$15
Key Caps	60	n/a	0
Straps	4	3.75	$15
		Total	$180

our target demographic.

Our goal was also to focus on a small subset of people with SH and build something for a very specific demographic. This is a subset that is rarely focused on and which would be the most affected group by our project. One approach for mitigating these issues is to have a very clear and precisely defined subset. Whilst SH is a small subset, it can manifest in people in different ways, and having a very clear and defined subset is very useful. At the same time however, a more human approach is also necessary. It would have been ideal to have essentially built the keyboard for one specific person, to at least show that the keyboard was capable of improving the lives of someone in our target demographic whilst not unintentionally over representing them.

Another issue involves the use of a RaspberryPi. Whilst there is no inherent moral issue with it, the Pi is a relatively complex device being used for a relatively simple function. A lot of the features in our Pi such as WiFi connectivity provides an additional and unnecessary security risk. In the future, it would be more practical to use either a simpler Pi, or a microcontroller that lacks many of the complex and potentially risky features present in our Pi. On top of this, a simpler device would also create a much more elegant solution which would not be as unwieldy, which would also improve our form factor.

10 RELATED WORK

There are currently many related accessibility input devices on the market. Since we plan our use-case to be video games (Minecraft) it is important to note Microsoft's adaptive controller. Like many of the alternatives currently available, it is a generalized controller that can be adapted to multiple disabilities. Our design differs from this in that we are focusing on a very specific subgroup which could benefit from a more niche input device.

There are a number of pre-existing devices which are often used by people with CP but which are not tailored to them. Examples of these devices include ball mouses, roller mouses, and one handed keyboards. Our device has the benefits of a one handed keyboard but with particular considerations for limited mobility, and common user experience problems for people with spastic hemiplegia.

11 SUMMARY

Our accessibility input device is a one handed keyboard with particular considerations for people with SH. This includes larger key sizes (20mm) as well as key guards which prevent unintentional input. We also include a mouse which can be controlled with one foot, allowing people to use the keyboard and mouse at the same time, a feature that is rare in the alternatives already available on the market.

Our challenges moving forward have to do with the particular challenges when designing for someone with SH. Despite the fact that our design focuses on the high functioning side of the body, the other side is prone to spastic behavior which can affect comfort and ease of use, even if our design is not intended to interact with that particular side.

It is important that we think of the device as a genuine improvement upon pre-existing alternatives and try to design something which is attentive to all the problems associated with CP and not just ignore the side of the body that is not in use.

Of all the things we had set out to do originally we did manage to hit many of our goals. We wanted to build a mouse and keyboard layout which could be used with only one side of the body and have options which would be unique and beneficial to those with Spastic Hemiplegia. This was our primary achievement but had we had more time we would have likely improved upon a few things in our design.

11.1 Things We Would Change

First and most importantly, we were unable to find input from someone with Spastic Hemiplegia during our design process, despite our attempts to find an informed advisor. Whilst we did receive very beneficial input from a Professor in the Human and Computer Interaction department, we would have liked direct contact with someone that could better inform us as to what features they would find more useful. This inability to find someone also stunted our testing step, which ended up being mostly done on CMU students that did not match our target demographic. Had we found a proper test group, we could have also added more unique software features to improve quality of life and mitigate errors much more efficiently.

Secondly, had we not had so many delays we would have likely refined and improved our housing and overall build quality. Currently it is not bad, but having key switches that are mounted straighter would have been beneficial and we could have also improved our housing to be much more robust.

Thirdly, the Raspberry Pi that we used was far more complex and had a lot of unnecessary features for our intended purpose. We did so as a contingency as we did not know how many features we would need but in the future it would be better to use a simpler Pi or a microcontroller that lacks the unnecessary bulk of ours.

Lastly, the keyboard layout could be rearranged. While testing showed that the letter keys were positioned well relative to each other, the supporting keys (spacebar, enter, punctuation, etc.) were awkwardly positioned. Testers could not see the supporting keys because they were covered by their hands. In the future, we will reposition the keys to be more intuitive.

11.2 Lessons Learned

Overall we did come up with an interesting and hopefully pleasant design for use with one side of the body which could be used for day-to-day browsing and application use. The ease with which we were able to make use of it with little practice also showed promise when considering whether someone could eventually make use of this as their primary keyboard for daily tasks.

I think it is important when approaching this problem in the future to have more contact with the target demographic. This was the sole issue in our project and could be improved upon greatly if this was the case.

Glossary of Acronyms

- CP - Cerebral Palsy
- RPi – Raspberry Pi
- SH - Spastic Hemiplegia

References

[1] *Input matrix scanning.* URL: `http://www.openmusiclabs.com/learning/digital/input-matrix-scanning/index.html`.

[2] Edmund F. LoPresti, Heidi Horstmann Koester, and Richard C. Simpson. "Measuring Keyboard Performance for People with Disabilities". In: 2006.

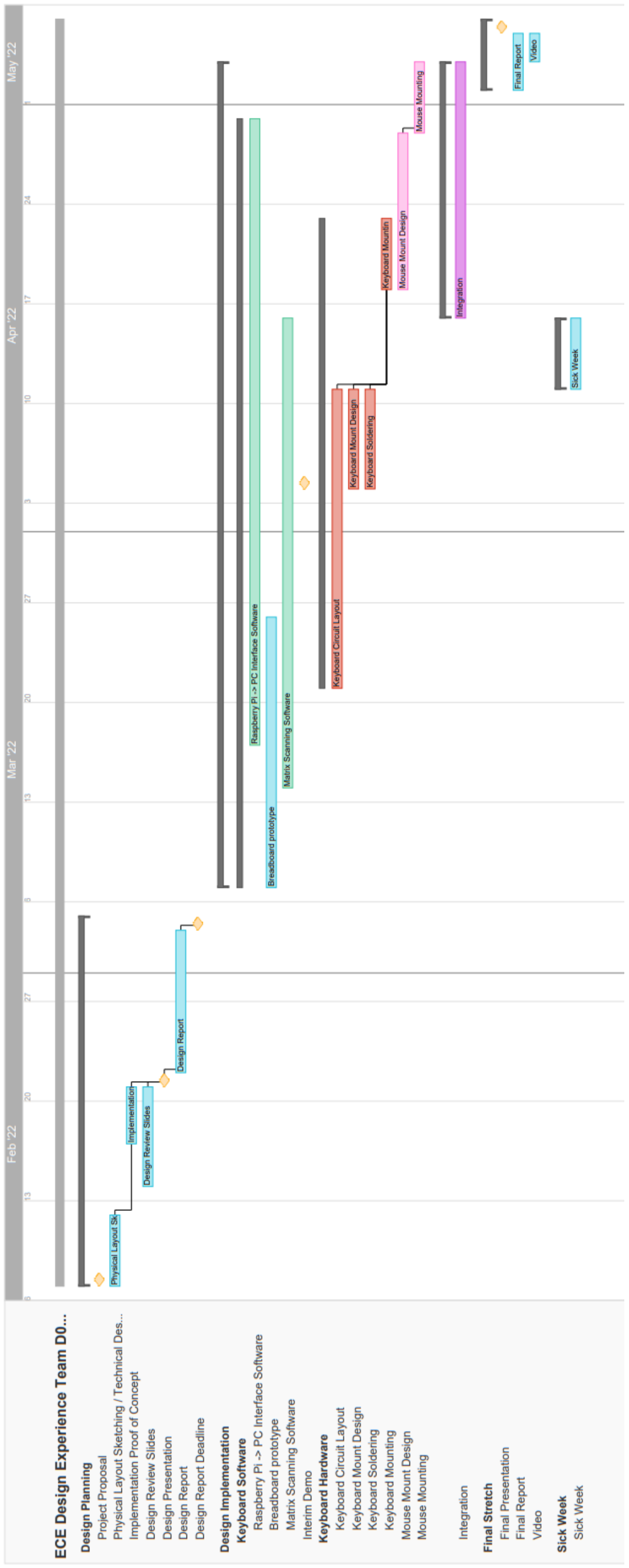

Figure 7: Gantt chart that details our final schedule for this project.